The Claims

1. (Currently amended) A software architecture for a distributed computing system comprising:

an application configured to handle requests submitted by applications executing on remote devices over a network; and

an application program interface to present functions used by the applications to access network and computing resources of the distributed computing system, wherein the application program interface comprises a set of base classes and types that are used in substantially all applications executing on the remote devices submitting requests, wherein the set of base classes and types comprises:

an AsyncCallback delegate supplied to an application, wherein the AsyncCallback delegate references a callback method to be called when a corresponding asynchronous operation is completed; and

an IAsyncResult interface that enables determination of the status of an asynchronous operation, wherein the IAsyncResult interface includes:

an AsyncState property that returns the object that was provided as the last parameter as part of a Begin call corresponding to the asynchronous operation;

an AsyncWaitHandle property that returns a WaitHandle that can be used to allow the application to wait for a call to be completed without needing to poll;

a CompletedSynchronously property that is set to true if the Begin call corresponding to the asynchronous operation completed synchronously; and

an IsCompleted property that is set to true after processing of the asynchronous operation is completed.

2. (Canceled).

3. (Canceled).

4. (Original) A software architecture as recited in claim 1, wherein the set of types support an event model including an event delegate that connects an event with a handler of the event, the set of base classes and types further comprising:

one or more classes that hold event data; and

one or more delegates that identify a method to provide a response to an event.

5. (Original) A software architecture as recited in claim 1, wherein the application program interface further comprises a collections namespace that includes a plurality of classes and interfaces for in-memory data storage and manipulation.

6. (Original) A software architecture as recited in claim 5, wherein the collections namespace includes, as at least part of the plurality of types:

a first set of types including commonly used collection classes;

a second set of types including interfaces to define a formal contract between developers creating new collections and developers consuming collections; and

a third set of types that support creating strongly typed collections.

7. (Original) A software architecture as recited in claim 1, wherein the application program interface further comprises a globalization namespace that includes a plurality of classes that define culture-related information, wherein the plurality of classes include a first set of types representing information about a user's culture and a second set of types representing information about a user's region.

8. (Original) A software architecture as recited in claim 1, wherein the application program interface further comprises a net namespace that includes a plurality of classes that enables use of network resources without details of one or more protocols used to access the network resources.

9. (Original) A software architecture as recited in claim 1, wherein the application program interface further comprises a security namespace that includes a plurality of classes and interfaces that make available an underlying structure of a security system including one or more cryptographic services, code access security and role based security infrastructure.

10. (Original) A software architecture as recited in claim 1, wherein the application program interface further comprises a service process namespace that includes a plurality of classes that allow installation and running of services.

11. (Original) A software architecture as recited in claim 1, wherein the application program interface further comprises a serialization namespace that includes a plurality of classes that enable serializing and deserializing of instance data.

12. (Original) A software architecture as recited in claim 1, wherein the application program interface further comprises a diagnostics namespace that includes a plurality of classes that enable debugging of applications, trace code execution, reading event logs, writing event logs, and monitoring system performance.

13. (Original) A software architecture as recited in claim 1, wherein the application program interface further comprises a messaging namespace that includes a plurality of classes that enable connecting to message queues on the network, sending messages to message queues, receiving messages from message queues, and peeking at messages from message queues.

14. (Canceled).

15. (Currently amended) An application program interface, embodied on one or more computer readable media, comprising: ~~as recited in claim 14~~

an AsyncCallback delegate supplied to an application, wherein the AsyncCallback delegate references a callback method to be called when a corresponding asynchronous operation is completed; and

an IAsyncResult interface that enables determination of the status of an asynchronous operation, wherein the IAsyncResult interface includes:

an AsyncState property that returns the object that was provided as the last parameter as part of a Begin call corresponding to the asynchronous operation;

an AsyncWaitHandle property that returns a WaitHandle that can be used to allow the application to wait for a call to be completed without needing to poll;

a CompletedSynchronously property that is set to true if the Begin call corresponding to the asynchronous operation completed synchronously; and

an IsCompleted property that is set to true after processing of the asynchronous operation is completed.

16-29. (Canceled).